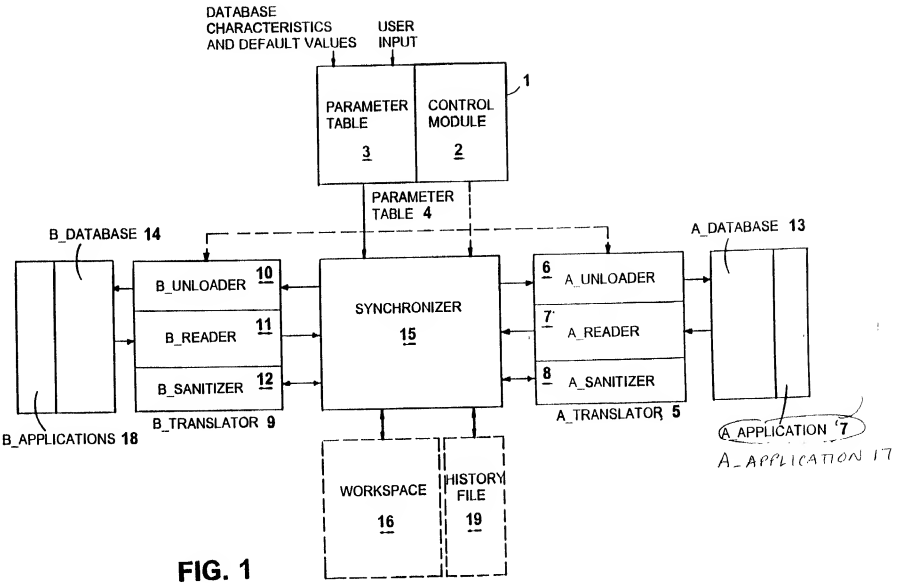
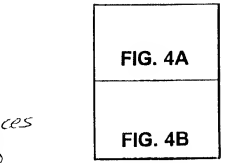


FIG. 1

Pseudo Code for Translation Engine Control Module

100. CREATE Parameter_Table from User Input A & B database characteristics and default values
101. INSTRUCT Synchronizer to initialize itself
102. INSTRUCT Synchronizer to LOAD the History_File into its WORKSPACE
103. INSTRUCT B_Translator to LOAD all of B_records from B_Database and SEND to Synchronizer (Synchronizer STORES these records in WORKSPACE)
104. INSTRUCT A_Translator to SANITIZE B_records that were just LOADED (A_Translator USES Synchronizer services to read and write records in the WORKSPACE; Synchronizer maps these records using the B→A_Map before sending them to A_Translator and maps them back using A→B_Map before rewritting them into the WORKSPACE)
105. INSTRUCT A_Translator to LOAD all of A_records form A_Database and SEND to Synchronizer (Synchronizes STORES these records in WORKSPACE by first mapping them using the A→B_Map and them storing in their new form)
106. INSTRUCT B_Translator to SANITIZE A_records that were just LOADED (B_Translator uses Synchronizer services to read and write records in the WORKSPACE)
107. INSTRUCT Synchronizer to do CAAR (Conflict Analysis And Resolution) on all the records in WORKSPACE.
108. INFORM user exactly what steps Synchronizer proposes to take (i.e. Adding, Changing, and Deleting records). WAIT for User
109. IF user inputs NO, THEN ABORT
110. INSTRUCT B_Translator to UNLOAD all applicable records to B_Database.
111. INSTRUCT A_Translator to UNLOAD all applicable records to the A_Database.
112. INSTRUCT Synchronizer to CREATE a new History File.

FIG. 3



Pseudocode for Generating Parameter Table

{Get Input from the user}

150. ASK user to whether to synchronize based on a previously stored set of preferences (Previous_Preferences) or based on a set of new preferences (New_Preferences)

151. IF New_Preferences THEN

152. ASK user whether Incremental Synchornization or Synchronization_from_Scratch

153. ASK user following information and STORE in Parameter_Table

a. A_Application and B_Application Names

b. ADB and BDB Names

c. ADB and BDB Locations

d. Which sections to Synchronize

e. Conflict Resolution Option: IGNORE, ADD, DB WINS, BDB WINS, or NOTIFY

f. Other user preferences

154. ASK user whether wants default mapping for the selected sections of the two databases or wants to modify default mapping

155. LOAD A_Database→B_Database (2)

156. IF Default_Mapping THEN

157. STORE A→B_Map AND B→A_Map in Parameter_Table

158. END IF

159. IF Modified_Mapping THEN

160. DISPLAY A→B_Map and B→A_Map

161. ASK user to modify Maps as desired

162. STORE the new A→B_Map and B→A_Map in the Parameter_Table

163. END IF

165. END IF

FIG. 4A

166. IF Previous_Preferences THEN
167. ASK user whether Incremental Synchornization or Synchronization_from_Scratch
168. STORE in Parameter_Table
169. LOAD Previous Preferences regarding which databases, mapping, and so on
170. STORE in the Parameter_Table
171. END IF

{User now specifies Date Range}

172. ASK user to choose Date Range Option
    a. Previously chosen Automatic_Date_ Range calculated from today
    b. Input New Automatic_Date_Range
    c. Input static Date Range for this Synchronization
    d. All dates
173. CALCULATE Start_Current_Date_Range and End_Current_Date_Range based on values from step 171
174. STORE in Parameter_Table
175. LOAD parameters setting out characteritics of A_Database and B_Database from Parameters database, including
    a. Field_List_A and Field_List_B
    b. A_Translator and B_Translator Module Identifiers
    c. ADB_Section_Names and BDB_Section_Name
176. STORE in Parameters Table

FIG. 4B

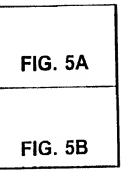


FIG. 5

200. RECEIVE following from Parameter Table
1) Name of A_App
2) Name of B_App
3) Name and Location of A_DB
4) Name and Location of B_DB
5) Section name of A_Application to be synchronized
6) Section name of B_Application to be (sy) synchronized
7) Incremental_Synchronization or Synchronization_From_Scratch Flags
201. SEARCH for H_File matching Parameters 1-6 / Synchronization
202. If Found H-File and Incremental_Synchrnization THEN DO nothing
203. IF Found H-File and Synchrnization_from_Scratch, THEN DELETE H_File
204. IF NOT found H-File, THEN SET Synchronization_from_Scratch AND ASSIGN file name for history file.
205. LOAD from Parameter_Table Start_Current_Date_Range and End_Current_Date_Range
206. LOAD from Parameter_Table Field_Lists for A-DB and B-DB and field and mapping information
207. If Incremental_Synchronization THEN COMPARE Field_Lists and Maps from Parameter_Table with History_Field_Lists and Maps
208. IF exact match THEN DO nothing
209. IF not exact match THEN DELETE H_file AND SET Synchrnization_from_Scratch
210. CREATE WORKSPACE using Field_List_B
211. If Incremental_Synchronization THEN Copy H_file into WORKSPACE
212. FOR each H-Record update
{analyze & update source of extended index}
213. Do Nothing to NEXT_IN_FIG
214.

FIG. 5A

Pseudocode for Key_Field_Match

```
250.    RECEIVE Key_Field_Hash and WORKSPACE_ID
251.    For all records in WORKSPACE
252.            IF Match_Hash_Value equals Hash Values of Record THEN LOAD the two records
253.                    COMPARE the key fields two records
254.                    IF Exact Match THEN SET Match_Found
255.                    EXIT LOOP
256.            END IF
257.    END LOOP
258.    If Match_Found THEN SEND Success Flag and WORKSPACE ID of Matching record
```

FIG. 7

Pseudo Code for Loading Records of B_database into WORKSPACE

B_Translator:

```
300.  FOR ALL Records in B_DB
301.        READ Record from B_DB
302.        IF (record outside of combination of Current_Date_Range and Previous_Date_Range), THEN
            GOTO END LOOP
303.        IF NOT right origin tag for this synchronization THEN GOTO END LOOP
304.        SEND Record to Synchronizer 325-236
305.  END LOOP
```

Previous

Synchronizer:

```
325.  RECEIVE B_Record
326.  STORE in WORKSPACE in next available space
```

FIG. 8

Pseudocode for Conflict Analysis And Resolution (CAAR)

500. Analyze ID_Bearing FIGS.
501. Analyze and expand ID_bearing CIGs
502. Finding Matches between Recurring Items and Non-Unique ID beaing Instances
503. Analyze SKGs
504. SET CIG Types

bearing

FIG. 12

Pseudocode for Analyzing ID_bearing FIGs

```
550.  FOR EVERY Recurring Master of ID_Bearing FIGS in H_file
551.      FOR EVERY FIG H_Record in Recurring Master FIG
552.          REMOVE Record from SKG it belongs to
553.          IF Record is a singleton CIG, THEN ADD to New_Exclusion_List
554.          IF Record is a doubleton CIG, THEN
555.              IF the two Records in CIG are identical, THEN remove other RECORD from
                  its SKG
556.              END IF
557.              ELSE IF the two records are NOT Identical, THEN ADD FIG record to
                  New_Exclusion_List and change records into singleton CIGs
558.          END IF
559.      END LOOP
560.      CREATE Synthetic Master record entry in WORKSPACE
561.      COPY value from one of the CIG mates into Synthetic Master
562.      COPY Rep Basic (i.e. recurrence pattern) from the Recurring Master into Synthetic Master
563.      COPY Exclusion List from the database Recurring Master into Synthetic Master and MERGE
          with New_Exclusion_List
564.      COMPUTE all Hash values for Synthetic Master
565.      CREATE new FIG between Synthetic Master the CIGmates of the H-FIG records
566.      CREATE CIG among the three Recurring Masters

{Fan Out Creep}                                   Previous
567.      Fan out Recurring Master with Previous Date_Range
568.      Fan out Recurring Master with Current_Date_Range
569.      IF two date arrays are NOT identical, THEN MARK CIG with Fan_Out_Creep flag
570.      MARK all Records in H_File Recurring Master FIG and Synthetic Master FIG as
          Dependent_FIG
571.  END LOOP
```

FIG. 13

Pseudo Code for EXPANDING ID_BASED CIGs

```
600.  For each H_record,                                   Dependent
601.        IF single record CIG, THEN GO TO END LOOP
602.        IF triple record CIG, THEN REMOVE CIG records from their SKGs
603.        IF Dependant_FIG, THEN GO TO END LOOP
604.        IF record needed to make triple has to be from a DB with unique ID, THEN GO TO END
            LOOP
605.        For all members of SKG to which H_record belongs
606.              IF Non_Key_Field_Hash of H_record and SKG_record Match, THEN
607.                    IF Exact Match of all fields with H item THEN Strong_Match is found END
                      IF
608.              ELSE
609.                    IF H_Record is a Recurring Master, THEN Find Fanned Instance (Table
                      Recurring Master/Instance Match) which is Strong_Match
610.              END IF
611.        END LOOP
612.        IF Strong_Match is found AND IF the SKG_Record is Weak_Match member of a CIG, THEN
613.              REMOVE SKG Record from Weak_Match CIG AND Seek Alternate Weak_Match for
                  the CIG
614.              ADD SKG record to Current doubleton CIG AND Record for the Weak_Match_CIG
615.              REMOVE all records in CIG from SKG
616.        END IF
617.        IF Strong Match is NOT found, THEN FIND Weak_Match
618.        IF Weak Match is found, THEN create Weak_CIG
619.              ELSE REMOVE all records in CIG from SKG
620.        END IF
621.  END LOOP
```

FIG. 14

Pseudo Code for Finding Weak Matches for a Record

```
622.  FOR EVERY Record in SKG
623.        IF (SKG record is from same database as records for which match is sought OR
624.              SKG record already is a Weak_Match record in a CIG OR
625.              SKG record is a Dependent_FIG OR
626.              SKG record is Non_Recurring AND records for which is sought are not, OR
627.              SKG record is Recurring AND records for which is sought are not)
628.        THEN
629.              GO TO END LOOP
630.        ELSE  If recurring
631.              Ifrecurring item OR Key_Date_Field match Exactly, THEN Weak_Match is found
632.        END IF
633.  END LOOP
```

FIG. 15

```
920.        IF Outcome = ADD, THEN
921.              GET Current values of all Fields, from Synchronizer
                  (Synchronizer maps for A database based on B→A, in response to each request)
922.              CREATE new RECORD in DB
923.              IF Unique_ID DB, THEN GET Unique_ID
924.              SEND to Synchronizer (Success FLAG with any Unique_ID) OR (Failure Flag)
925.              Synchronizer: Store Unique_ID in WORKSPACE
926.        END IF
927.        IF Outcome is UPDDATE THEN GET Current values to be unloaded and original values loaded
            from database from Synchronizer
928.              COMPARE and DETERMINE which Field to be updated
929.              UPDATE fields in the record to be updated
930.              SEND to Synchronizer (Success flag AND Unique_ID) OR (Failure Flag)
931.              Synchronizer:  STORE Unique_ID in WORKSPACE
932.        END IF
933.  END LOOP
```

UPDATE

FIG. 25B

1050. Verify History File
1051. If verified, Then Proceed as Fast Synch
1052. If not, Then Proceed as Synchronization from Scratch load all record in databasse

1053. If Fast Synch
1054. LOAD records into the Workspace. Map if necessary
1055. Sanitize Records not marked as Deletion
1056. Orientation analysis (Fig. 11).
1057. For each H_Record, analyze the CIG that the H_Record belongs to.
1058. IF the H_Record's CIG contains no Record from the Fast Synchronization database, THEN CLONE the H-Item, label it a Fast Synchronization Record, and add it to the H_Record's CIG.
1059. If the H_Record's CIG contains a Fast Synchronization record that is marked as a Deletion, it is now removed from the CIG.
1060. If the H_Record's CIG contains a non-Delete Fast Synchronization Record, then do nothing.
1061. END LOOP

FIG. 30

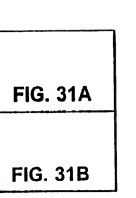


FIG. 31

1150. Verify History File
1151. If verified, Then Proceed as Fast Synch
1152. If not, Then Proceed as Synchronization from Scratch

1153. IF synchronization from scratch
1154. IF record outside of current_date_range THEN MARK record as out-of-range

1155. If Fast Synch
1156. Load History File into Workspace
1157. MARK History File records outside of previous_date_range as Bystander
1158. Load All Fast Synchronization Records into the Workspace; mapped if necessary.
1159. SANITIZE Records which are not DELETEs
1160. Orientation analysis (Fig. 11). Synchronization
1161. If Added Fast Synchrnization record is out of current date range THEN MARK Out-Of_Range
1162. If Changed or deleted Fast Synchronization record in a CIG with Bystander H_Record, MARK the Bystander record as Garbage

FIG. 31A